The Requirements Domain for Laboratory Software Infrastructure. RTLabOS: Phase I – Deliverable 1.1

Kosek, Anna Magdalena; Heussen, Kai

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

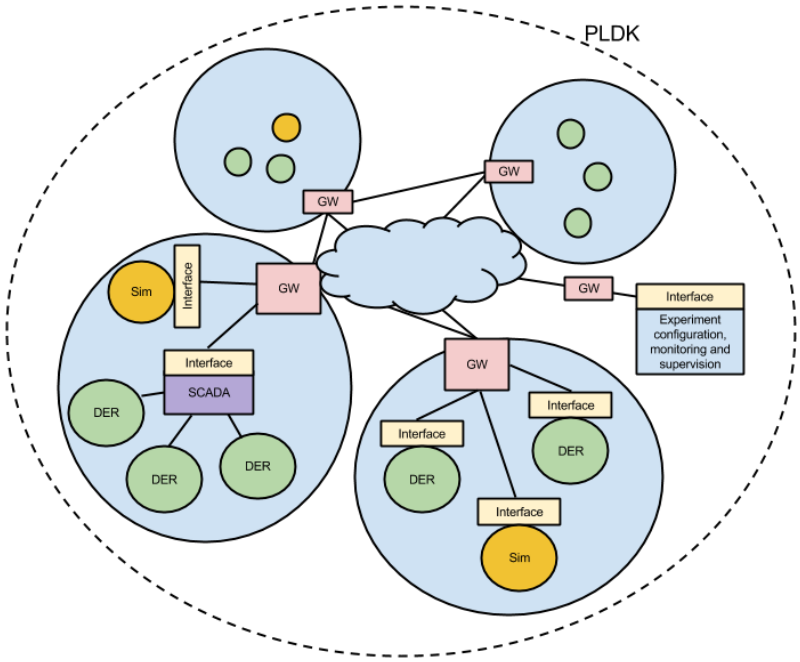
The Requirements Domain for Laboratory Software Infrastructure

RTLabOS: Phase I – Deliverable 1.1

Anna Magdalena Kosek, Kai Heussen

Technical University of Denmark

October 22, 2013



Contents

1 Introduction

Laboratories in a Smart Grid context are moving toward further integrated systems, where the complexity and software intensity of technologies is increasing. A cornerstone of effective and state of the art work in this field is the software infrastructure to operate the lab and support its key activities.

Many software technologies that support lab related activities, such as analysis and development, experimentation and testing, demonstration and education, are available today, and tailored solutions are in use in each laboratory. In practice, the complexity of (often changing) requirements and increasing capabilities, as well as the weight of legacy systems need to be balanced. Whereas some convergence in interoperability and standardization can be observed, the testing and development needs that are to be supported by research laboratories in the electric power domain are advancing to address full systems integration.

As opposed to hosting a static, single-vendor solution, research labs need to offer the flexibility to support integration of multiple vendors and the capability to offer an experiment-managing infrastructure on top of the subject of testing. To maintain a software infrastructure in such a setting that is state-of-the-art, affordable, free of vendor lock-in, flexible and configurable, easy to use, extensible and evolvable, all at once may be an impossible expectation.

In face of this challenge, we may ask: What are the key features that drive specific solutions? Can we identify common interfaces, architectural patterns, or good practices across laboratories? Are there shared requirements that could benefit from cross-laboratory collaborations?

The goal of RTLabOS: Phase I is to identify the requirements and key features for a next generation of smart grid lab software support. The following vision defines a direction for this development:

RTLabOS vision *The vision of RTLabOS is a supportive, real-time, cross-location laboratory software infrastructure for development and testing of topology independent and system-wide controls. Such an infrastructure will allow for seamless integration of simulated and physical components, and support open-platform- and standards-oriented development of solutions that can easily be deployed in the real world, enabling simulation and experiments with all relevant time-scales. Designed with all phases of experimental development in mind, it will offer support starting from experimental set-up and configuration, through online supervision and monitoring, to the tracking of relevant data-sets from various sources. It will be based on a software architecture that strikes the balance between ease of access, meaning low-entry threshold and simple configuration, and the flexibility needed for laboratory software which is under constant development.*

1.1 Purpose

This report is part of the state of the art and requirements analysis which aims for an understanding of the basic trade-offs driving lab software choices, as well as to avoid a conceptual lock-in to existing laboratory software systems. The purpose of this document is to define a common basis for classifying the challenges addressed by one way or another of conceiving a lab software infrastructure.

Within this report, the basic notions of the user requirements and the supported software are outlined.
This report forms a background for the RTLabOS Survey (Deliverable 1.2).

1.2 Scope

The scope of this report is the domain of the power system laboratory software infrastructures for development and testing of control strategies. In this report we focus on relevant lab users, activities and software tools.

In particular, we aim to identify practical aspects of conducting experiments in several lab facilities, possibly enabling remote access and monitoring. The domain of lab experiment can be divided into several parts:

- stakeholders - owners and operators of the lab
- users - using and contributing to the lab,
- activities - that can be performed in the lab,
- software tools- to use, maintain and observe the lab.

Section 2 focusses on the stakeholders and activities, Section 3 presents overview of lab software tools types. This report presents a domain study and provides basic concepts and terms for survey in [6] and survey questionnaire in [7].

2 The Laboratory Environment: Stakeholder Preferences and Requirements from Activities

The requirements of the lab software infrastructure ultimately depend on the use of the lab, its purpose and users requirements. To provide orientation, this chapter aims to structure these non-technology aspects of the requirements. The different parties interested in labs are mapped in Section 2.1, the focus on the Section 2.2 is on the actual lab users; the possible lab use and activities are presented in Section 2.3. Finally, the finally some overarching requirements aspects are addressed in Section 2.4.

2.1 Stakeholders

Stakeholders are all parties involved in proposing, planning, funding, hosting, operating and using a lab facility. The interests of these stakeholders are reflected in the way funding for a lab is strucured and which aspects of operation are prioritised.

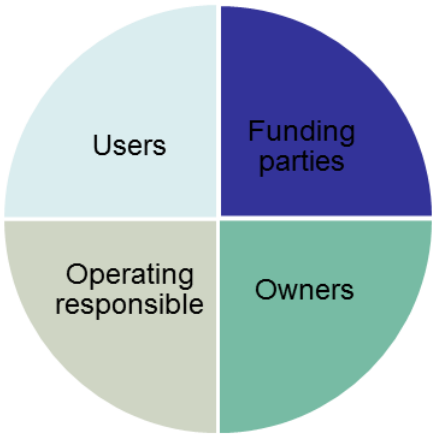


Figure 1: *Stakeholders of a lab facility.*

Funding parties Funding parties of the lab can include government institutions, investors, companies, universities.

Owners Ownership of a lab can be shared among different parties. Typically the owner is an institution that hosts the lab facility, for example university, national laboratory, or a company.

Operating responsible The entity *responsible* for operation of the lab, typically making decisions about the lab are the steering committee and the board of advisors.

Users The group of actual users of the lab; this category is further discussed in Section 2.2.

2.2 Lab Users

The requirements derived from actual lab use depend the specific role and interest of the user group. Let's consider the groups of stakeholders using lab facilities and map their interests, as presented in Figure 2.

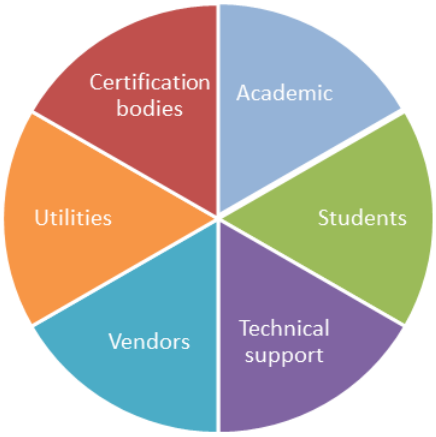


Figure 2: *Different parties using lab facilities.*

The lab users are classified into six groups. The group description and their main interests are described below.

1. **Academic** users are mainly focused on research and education. The lab can be used to perform experiments, test new algorithms, control and optimization methods and gather data for analysis. Educational use of lab can include demonstrations and lab use for academic courses and student projects. Whereas flexibility is a key interest for research purposes, in particular the education perspective promotes ease of access and re-use.
2. **Students** can use lab as a basic of their projects or academic courses, for gathering data, learning methods, exploring new algorithms and ideas. In labs with equipment students can familiarize themselves with operation and safe use of machines. Modern equipment, competent and available research and educational staff are most desired by this stakeholder.
3. **Technical support** maintains and monitors equipment in lab facilities. Technical support can also offer help to other lab uses by performing testing and validation. For technical support staff, robustness, overview and repeatability of procedures are priority values.

4. **Vendors and supply industry** users are mainly focused on testing products and validating their operability, as well as lab demonstration of developed products. Intellectual property issues need to be addressed such as data storage and transfer. Unique capabilities, repeatability of tests and an efficient business process for testing techniques are salient lab features for this stakeholder group.
5. **Utilities** use labs for pre-deployment testing and characterization of new technologies and to support exploration of future technology development, including, equipment, controls and operator support tools.
6. **Certification bodies** use labs for testing equipment for compliance with standards and regulations; also the generation of experience feeding into the development of new standards can be of interest to this stakeholder.

2.3 Activities

Lab use can be divided into eight activities: demonstration, experimentation, testing and validation, time series acquisition, decision support and tool development, maintenance and monitoring of equipment, controller development and deployment, and system engineering, as presented in Figure 3.

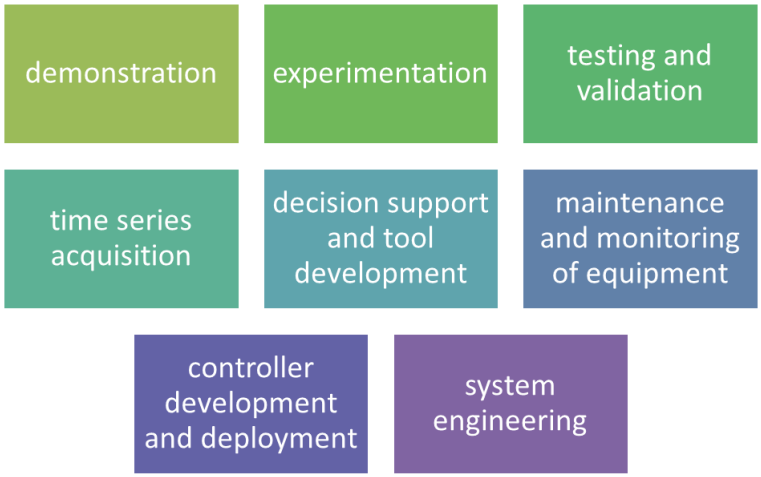


Figure 3: *Different activities in lab use.*

Demonstration A lab can be used to demonstrate an existing solution that can be deployed in the near future, or a proof-of-concept in the controlled lab environment. The purpose in either case is demonstration of capabilities and features of the solution on a realistic but controlled lab set-up with use of equipment that represent operations in the real system.

Experimentation For the laboratories in consideration here, the main purpose is to support scientific research and thus to accommodate experiments for testing ideas. Within the (typically narrow) scope of such experimentation, usability and configurability by the academic and

research staff with minimum dependency on specialist technical staff is desired. Experiment logging, user and operator interfaces are needed to the extend that they facilitate the execution of experiments.

Testing and validation A lab can be used to perform testing and validation of commercial products that are off-the-shelf or near-market, testing if devices comply with regulations and standards; work as expected individually and in combination with other equipment.

Time series acquisition Time series of the operation of a lab equipment can be gathered for a statistical analysis, a visualization, can be an input to controllers and be a basis for creating and validating models. In order to develop models of lab components, time series of the unit operation must be available along with information of unit control configuration and operating state. The lab software infrastructure can also facilitate time series- and model-sharing and reuse.

Decision support and tool development Tools for operators, grid monitoring and operation need to be a part of the software infrastructure supporting a lab facility. These tools are used on the one hand for operation of the lab itself and demonstration purposes, but they can also be subject of research activities themselves.

Maintenance and monitoring of equipment Scripts can help automating the maintenance of hardware equipment. In case of software maintenance a remote access and monitoring tools can be present. Tools enabling advanced units monitoring, able to run maintenance scripts for equipment fault detection, is needed for smooth and safe operation of lab facilities.

Controller development and deployment Controls development requires research in control design methods, new control structures, novel controllers and control policies, testing and tuning of controllers for deployment. Tools for controller development, deployment, execution and testing are needed to support academic staff and students (for example simulation models of different levels of refinement, controllable assets). This also include development of a design cycle, process planning, and tools for combing simulation models with controllers and lab equipment (co-simulation. hardware-in-the-loop, software-in-the-loop).

System engineering For labs with many power system units, system integration research, development and testing work can be performed. The software infrastructure supporting this activity can provide tools for system architecture design, and provide library of tools for system integration (for example common interfaces and protocols, deployment of supervisory controllers, monitoring and data collection). System engineering can also be associated with coupling of simulation environments.

2.4 Other Non-technical Factors

From the requirements perspective, Sections 2.2 and 2.3 provided some categories for prioritizing lab requirements. Further distinction criteria can be conceived to map out the activity-centric features of lab software infrastructure. In this section additional dimensions of activities presented in Section 2.3 are considered, including: complexity of experiments, software implementation effort, complexity and change, interoperability and communication.

Complexity of experiments The software infrastructure can support different complexity of the system or a set-up of the experiment. The size of the experiment or a demonstration is one dimension: starting form a component test to larger system integration and aggregation experiments. Along the same lines, one should consider the complexity of the system under test: starting from a homogeneous set of units to heterogeneous, cross-layer, cross-domain systems (for example co-simulation, hardware-in-the-loop).

Software implementation effort The infrastructure needs to provide tools that shorten development time and decrease the software development effort. Easy deployability of controllers, experiment set-up, including tools enabling interoperability, hardware/software co-simulation tools.

Complexity and Change The processes and modules that are building blocks of the software infrastructure supporting labs operation, can be loosely structured or highly formalized. The development of such formal building blocks is often driven by internal organization needs. Formal structures reduce complexity, however, their maintenance also requires significant resources in face of frequent change. Formal building blocks or structures can be developed with respect to:

- Re-use of models made in different tools, by providing model translators or interfaces.
- Re-use of controllers, controllers compatibility (possibility to merge many controllers in a single system), common interfaces for control (interfaces between aggregations in many levels and interfaces to operated units).
- Data interoperability, consistency, cataloging and storing.
- Standardization of simulation tools, their interoperability with different simulation or hardware tools.
- Data harmonization and merging, data compatibility, meta-data representation and classification (e.g. describing data validity).
- Common formats for data sharing and re-use, common data model translators.
- Formalized process for experiment set-up and maintenance, possibly automated.
- Ability to include new and remote assets, remote control, including data services (e.g. weather forecast service).
- Access control to assets, authorization, authentication and security.

Interoperability and Communication The functionality provided by software infrastructure covers several layers of application, from basic process communications and PLC programming, planning and decision support. Interoperability of software is often tied to a specific layer or from one layer to the next. In a laboratory context, the focus of research on a particular layer determines what type of solution is preferred: existing software components, new software off-the-shelf or developed as part of several projects, etc. An example of an automation architecture consisting of different layers is presented in Figure 4. Another related layer model is the current standardization work regarding the Smart Grid Architecture Model (SGAM).

- Upper Layer: 'business IT', including information management
- Scheduling and planning
- SCADA and Operator support
- Substation control level
- Component and asset control

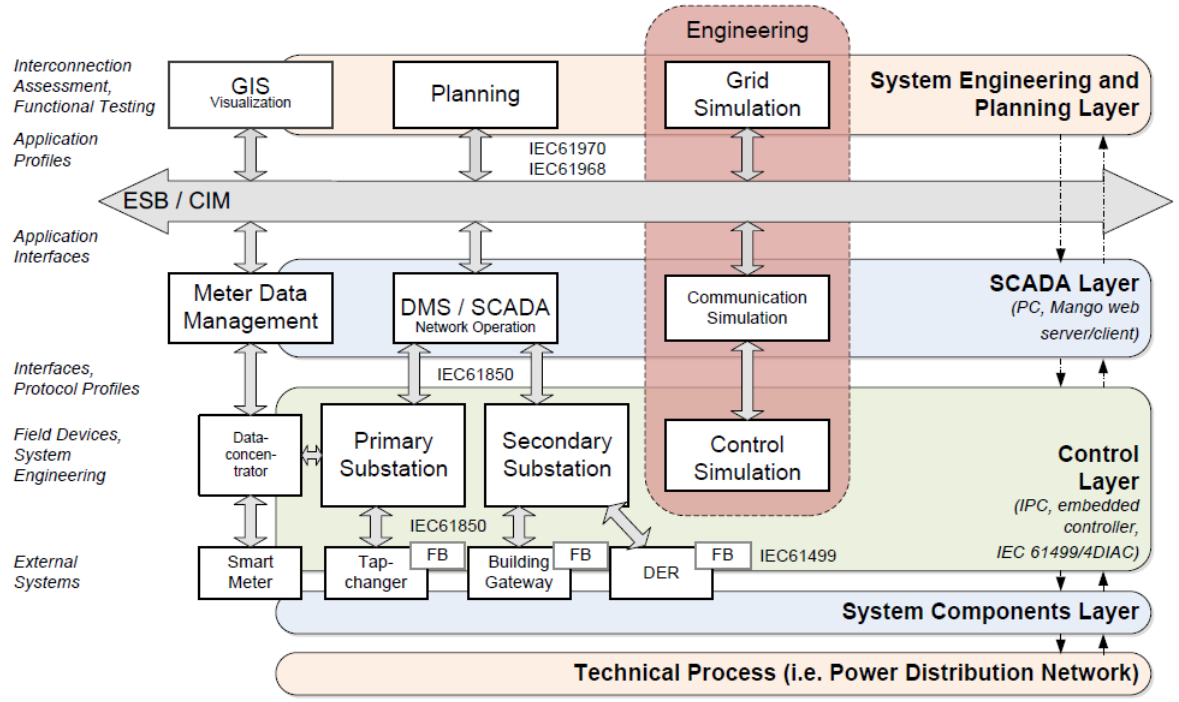


Figure 4: *Automation architecture layers with exemplification of associated standards. In a laboratory context the research focus will determine at which layer fixed solutions are preferred and where in-house solutions play a role.* Source: Thomas Strasser, AIT

- asset protocols (e.g. RS.232)

Relevant aspects to be considered in the context of interoperability and communication of the software infrastructure for labs are: relevant protocols and standards,definition of the scope of the laboratory, design and implementation focused on standards compared to custom interfaces and protocols, future standards and trends to be considered. These aspects has to be defined in the requirements of the RTLabOS software infrastructure, only selected solutions can be implemented, but the overview of the existing standards and their usability for a lab software infrastructure need to be assessed.

3 Software Tools

In this chapter examples of types of software tools that can be used together with the RTLabOS software infrastructure are described. These software tools are grouped into tools with an indirect relation to the lab (such as analysis and modeling tools), and directly lab-related tools. The researcher tools are mainly used for setting up, executing, observing and analysing experiments, and lab use tools consider monitoring, maintenance and operability of the lab infrastructure.

3.1 Analysis and Development Tools

The software tools associated with this section have an indirect association with the physical lab. Such tools are focussed to support the primary activities of research-oriented staff. This includes

use of models, algorithms and control, decision-support and visualization tools, simulation software, collaboration tools, data analysis and merging tools, and code management and development environments.

Models Models are important part of working with simulation and hardware-in-the-loop. Publicly available or custom models from electrical, mechanical, economical, meteorological domains should be easily accessed, and deployed in the chosen environment. Models should be stored in a common repository, available to all authorized parties. Models can come form public repositories, commercial products, projects and academic courses. Tools enabling easy reuse of existing models in experiments and simulations brings a desirable functionality for researchers and students.

Data for analysis Tools to access, select and import data are needed for a lab software infrastructure. Automated translation from different data types to a common format can speed up the data analysis stage. Tools to analyze, plot and combine results can be helpful for visualization, monitoring of an experiment or a demonstration, and data analysis.

Algorithms and control Many of researchers work only with control algorithms, a infrastructure for easy testing and simulation of algorithms is required to help evaluate their behaviour in the controlled system. Tools to deploy these algorithms in the lab are also needed for tests with actual power system components, communication network and users. Software infrastructure for deploying controllers is able to create containers where controllers can be deployed. Distributed controllers can be interconnected and deployed on different machines. Controllers can access other software infrastructure tools (e.g. database systems) and output data to GUIs and visualizations. Software infrastructure can provide interfaces between controllers and equipment, and also simulation tools involved in an experiment.

Decision-support and visualization tools Many lab activities are more efficiently executed when the operation of the power system or its components can be observable. Visualization tools helps in setting up and running the experiment, in the demonstration process - showing how the solution behaves in the lab environment.

Software licenses If a commercial software is used, software licences should be stored in a common space and easily accessed by the users.

Simulation software Simulations are mainly use to shorten the time of development of algorithms, before lab tests and field tests. Simulations can be also used for up-scaling, models of the power system components can substitute for units that are lacking in the lab (software-in-the-loop). The co-simulation concept uses many simulations to simulate a complex domain, for example influences between power system, communication and user domains.

Development environment Simulation software and tools for deployment of controllers and algorithms can be considered development environment, where control and aggregation concepts can be tested and further developed with use of software and lab facilities.

Information access and document management Some information in the lab should be shared and easily accesible, for example time series from renewables, unit configuration. All data from commercial tests or demonstrations should be kept confidential, and be stored in a secure database or repository.

Code management The lab software can also be the subject of research or experiment. If there is many developers working on a single software solution, a version control system is required, to track all changes. Version control software offers tools for code management, change and developer tracking and offers roll-back options. Additional tools to maintain software can be installed, for example bug trackers or scrum agile software development frameworks.

Collaboration tools A demonstration or an experiment can be conducted in a cooperation between several parties: researchers, companies or students. Information share, common experimet access gateways, monitoring and visualisation tools can be useful to jointly maintain and execute tests and conduct research. For new users of the lab wikis and how-tos are needed to familiarize themselves with the lab operation.

3.2 Lab Related Tools

In this section the domain relevant aspects of lab related tools are listed and functionality is clarified. Such tools support activities including experimental setup and configuration, on-line supervision and monitoring, and tracking of relevant data-sets from various sources. Selected topics considering lab related tools are: SCADA system, GUIs for monitoring and lab management,equipment and components, data aquisition and storage infrastructure,

SCADA system SCADA (Supervisory Control And Data Acquisition) enables monitoring, control and data aquistition from all lab facilities. Centralised SCADA gathers all information and send control commands to all supervised units form a single computer or a cluster (*example*: ABB SCADA is installed in the Intelligent Control Lab in PowerLabDk [8]). A distributed SCADA system is deployed in several computers, configuring and gathering data from a single or a group of DERs and exposing an interface to control the power system unit (or units) remotely (*example*: A distributed SCADA system is SYSLAB software, developed by DTU and deployed mainly in SYSLAB laboratory in Risø Campus of PowerLabDk [8]).

GUIs for monitoring and lab management Visualising the power grid and its components helps in maintenance of the equipment and can be a crucial part of the decision support an operator software tools; often it is an integrated aspect of a SCADA system.

Data aquisition and storage infrastructure Data acquisition requirements in a laboratory often exceed those from the field use. With a dedicated DA system, detailed evidence from experiments and tests can be collected, which also entails massive amounts of data to be handled. Data from measurement units need to be stored in an easily accessible and consistent repository or database. Data storage structure (text, binary files, database) need to chosen and implemented. Data about the experiment run, user and operator, can also provide information about activities happening in the lab.

Equipment and components control Controlling and monitoring equipment can be difficult to manage without suitable machine-to-machine interfaces. Such control requires simple, device-level interfaces that also support remote control and data acquisition tools, for example SCADA, are needed to support all lab activities.

3.3 Additional System Integration Tools

The tools presented in this subsection are advanced lab tools that are not required but can be interesting for the lab environment, mentioned here system integration tools cover: co-simulation and HIL coupling and interconnection, remote access to equipment and simulation, data merging, experiment booking and permissions, platform for deploying controllers are not required for standard lab operation but are required for advanced work with system integration

Co-simulation and HIL coupling and interconnection Lab software infrastructure should be able to track all simulations and equipment available for experimentation. Interfaces to equipment and simulations need to enable coupling, lab software infrastructure can provide means of translation between one interface or another (in sense of data model or message format translation). Co-simulation tools:

- mosaik - co-simulation orchestrator [3],
- FMI (Functional Mock-up Interface) proposing standardized interfaces for cyber-physical co-simulation [4],
- Ptolemy tool for simulating concurrent, real-time, embedded systems[5].

Remote access to equipment and simulations Equipment and simulations could be remotely configurable, started/stopped and controlled, data can be piped to an application or saved in a database using secure remote access.

Data merging and model translation tools Data formats, models, units and time-stamping vary between different systems. A way of translating and merging data to provide structured experiment data and transparent knowledge-sharing can be achieved by such tools.

Consistent experiment booking, permissions and data tagging A tool or infrastructure that integrates the process from managing the booking of lab elements, managing authorization and authentication, as well as associating recorded data with tags that relate to the relevant data.

Platform for deploying controllers Platform is able to create containers where controllers can be deployed. Distributed controllers can be interconnected and deployed on different machines. Controllers can access other platform tools (e.g. database systems) and output data to GUIs and visualizations. Platform can provide interfaces between controllers and equipment and simulation tools involved in an experiment.

Advanced configuration management Lab configuration can be done manually, but for more advanced experiments or demonstrations automatic configuration can be a useful tool. These tools may include: storing experiment configuration, automatic system reconfiguration, for example with use of ontology driven models, generic models, data fusion or techniques from artificial intelligence.

Service-oriented tool access Number crunching services, forecasts, GIS services, real-time grid data and state-estimation, load-forecasting and weather services, are all examples of data-related services that require bundling of specific data types with dedicated computation services. A software infrastructure that enables convenient API-based access to such services is certainly advanced.

Multi-agent system driven lab platforms Agent-oriented programming is an advanced software paradigm that associates a higher level of a autarky with software components, which can be useful to enhance adaptability, robustness and integration between lab components. As a strong programming paradigm, it also entails an overhead of programming concepts that users need to adopt. To develop multi-agent system already existing platforms can be used. Example of existing platforms are for example: open-source Jade, [1], commercial Jack [2]. Multi-agent simulator for co-simulation, with interface to mosaik [3] co-simulation coordiantor tool.

4 Conclusion

The domain of power system laboratory infrastructure encompasses a wide variety of aspects, arguably including aspects of power systems and those of running a research laboratory. The key to lab software support would thus be to focus tools on the specific needs and ombinations of lab activities rather than to aim for a generic solution.

Given the categorization provided here, identification of such needs should be possible using a survey on power systems laboratories, considering jointly, their currently applied software technology and the drivers that lead to their use.

Finally, the taxonomy of lab related software not only provides an overview, It also serves as inspiration for development opportunities.

References

[1] Jade - java agent development framework, 1999.

[2] Jack by aos autonomous decision-making software, 2009.

[3] mosaik by OFFIS Institute for Information Technology, 2011.

[4] Functional Mock-up Interface (FMI) by Modelica Association Project, 2012.

[5] Ptolemy II by UC Berkeley, 2012.

[6] Kai Heussen. RTLabOS D1.2 Survey. Technical report, Department of Electrical Engineering, Technical University of Denmark, 2013.

[7] Kai Heussen and Evgenia Dmitrova. RTLabOS D1.2 Survey Questionnaire. Technical report, Department of Electrical Engineering, Technical University of Denmark, 2013.

[8] Anna Magdalena Kosek. RTLabOS D1.3 PowerLabDK description. Technical Report RTLabOS D1.3, Department of Electrical Engineering, Technical University of Denmark, 2013.

Appendix A: Open Source Projects

It is apparent that a large number of open source initiatives aim to provide free and dependable tools needed in a laboratory and research environment. To provide an impression of this work, here an unsorted list of open source projects in the context of SCADA systems development:

- HMI
  - http://sourceforge.net/directory/science-engineering/hmi/freshness:recently-updated/
- Modeling and Simulation
  - http://sourceforge.net/projects/gridlab-d/?source=directory
  - http://sourceforge.net/projects/modelio-open/?source=directory
  - http://sourceforge.net/projects/modeliouml/?source=recommended
  - http://sourceforge.net/projects/jmcad/?source=directory
- Interfaces
  - http://sourceforge.net/projects/pyvisa/?source=directory
  - http://sourceforge.net/projects/openopc/?source=directory OpenOPC for Python
  - http://sourceforge.net/projects/opycua/?source=directory OPyCua (Prealpha) OPC UA communication stack written in python
- SCADA
  - http://sourceforge.net/projects/openscada/?source=directory
  - http://www.openapc.com/ OpenAPC – Open Advanced Process Control
  - http://sourceforge.net/projects/tango-cs/?source=directory http://www.tango-controls.org/
  - http://sourceforge.net/projects/sardana/?source=directory
  - http://sourceforge.net/projects/indigoscada/?source=directory
  - http://sourceforge.net/projects/seer2/?source=directory
  - Overview: http://sourceforge.net/directory/science-engineering/scada/freshness:recently-updated/
- Test and Measurement
  - http://sourceforge.net/directory/science-engineering/testmeasure/freshness:recently-updated/
- Technology
  - http://sourceforge.net/projects/opencvlibrary/?source=directory
  - http://sourceforge.net/projects/mrpt/?source=directory http://www.mrpt.org/
  - http://sourceforge.net/directory/science-engineering/ai/intelligent-agents/freshness:recently-updated/

In related fields of data analysis, programming and simulation, the list would be even longer. As an excerpt, here three open source tools that are so well established in their field that they are an easy match for their commercial counterparts.

- the *Eclipse* project
- *R* for statistical computations
- *Octave* for matrix-oriented programming
- *MatPower* for Power Flow calculations and OPF

The key benefits from open source, and the wider field of open API developments in this field are the dependability of transparent code and possibility for contributing special-purpose libraries, which for researchers serve the double-purpose of simplifying their work and communicating their results to a large community.